(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 578 205 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **93110797.3**

(51) Int. Cl.[5]: **G06F 15/403**

(22) Date of filing: **06.07.93**

(30) Priority: **06.07.92 US 909984**

(43) Date of publication of application:
12.01.94 Bulletin 94/02

(84) Designated Contracting States:
AT BE CH DE DK ES FR GB GR IE IT LI LU MC NL PT SE

(71) Applicant: **MICROSOFT CORPORATION**
One Microsoft Way
Redmond, Washington 98052-6399(US)

(72) Inventor: **Miller, Thomas J.**
3135 Fairweather Place
Bellevue, Washington 98004(US)
Inventor: **Kimura, Gary D.**
6807 126th Avenue N.E.
Kirkland, Washington 98033(US)

(74) Representative: **Patentanwälte Grünecker, Kinkeldey, Stockmair & Partner**
Maximilianstrasse 58
D-80538 München (DE)

(54) Multiple file name referencing system.

(57) A multiple file name referencing system stores multiple file names in a file. These multiple file names include an operating system formatted file name and an application formatted file name. When an operating system formatted file name is created or renamed, the multiple file name referencing system automatically generates an application formatted file name having a potentially different format from, but preserving the extension of, the operating system formatted name. The multiple file name referencing system similarly generates an operating system formatted name upon creation or renaming of an application formatted name. A B-tree is provided which contains an operating system entry for the operating system formatted name and an application entry for the application formatted name, each entry containing the address of the same file to which both names refer. The multiple file name referencing system converts the operating system formatted file name to the application formatted file name by accessing the B-tree with reference to the operating system entry, and vice versa. As a result, either file name can be used to directly reference the file without requiring additional file name translation.

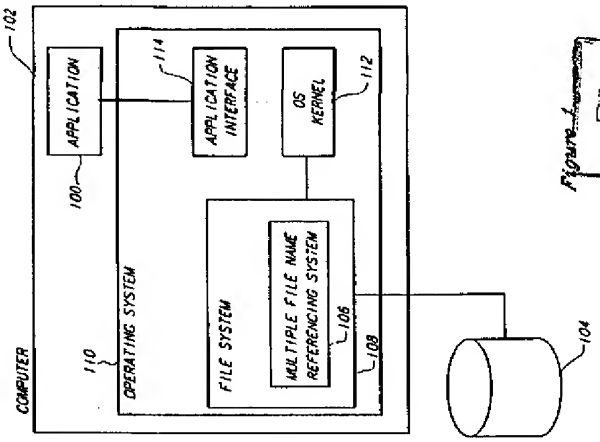


Figure 1

PH NL 030420 WO
MAT. DOSSIER

EP 0 578 205 A2

BNSDOCID: <EP_____0578205A2_I_>

Technical Field

The present invention relates to the field of file systems and, more specifically, to a multiple file name referencing system by which a file can be referenced by multiple file names of different formats.

Background of the Invention

Previously, applications implemented on a typical computer system have had to reference file names using the same format as the underlying operating system provides. For example, the DOS operating system (developed by Microsoft Corporation) internally references files with a format that corresponds to an "8.3" formatted file name. That is, DOS would use a file name having a leading name of up to eight characters, such as "myfile" or "filename", followed by a period and an extension of up to three characters, such as "exe" or "kp". Accordingly, applications running on DOS have had to limit file names to this format. Additional constraints well known in the computer field apply to DOS file names, and will be described in more detail in the Detailed Description of the Invention.

In the computer industry, however, it is not uncommon for improvements to be made to an existing operating system, or for all or a portion of an operating system to be replaced so as to provide greater capabilities to users of the operating system. For example, an existing operating system may be revised to allow for files to be names with a more flexible format. It thus becomes possible for applications running on such a revised operating system to reference files using the more flexible format.

When an application is written to run on the revised operating system, the application may access and name files using the more flexible format. However, a number of applications remain available in the industry which were written to run on the previous operating system. As such, these applications impose the same constraints on file names as did the previous operating system. For example, an application designed to run on DOS would limit file names to the "8.3" format described above. As a result of this limitation, the application could not reference files using file names of the more flexible format of the revised operating system. Thus, the application could not refer to files already named with the more flexible format. Nor could the application itself cause new files to be created for which the revised operating system could take advantage of the more flexible format in referencing the files.

Summary of the Invention

An object of the invention is to provide a method and means for referencing the same file via multiple file names of varying file name format.

This object and other objects of the invention can be obtained by a multiple file name referencing system described as follows. The multiple file name referencing system is provided within the file system of the revised operating system and stores, as necessary, multiple file names in a file. These multiple file names include an "application formatted" file name which is formatted for applications written to run on a different operating system, and an "operating system formatted" file name which is formatted to run on the revised operating system. When an operating system creates or renames a file having an operating system formatted file name, the multiple file name referencing system automatically generates an application formatted file name of a different format from the operating system formatted name. Similarly, the multiple file name referencing system automatically generates an operating system formatted name upon user creation or renaming of an application formatted name.

The multiple file name referencing system maintains in a B-tree an operating system entry containing the operating system formatted name and an application entry containing the application formatted name. Each entry also contains the address of the same file to which both names refer. The multiple file name referencing system converts the operating system formatted file name to the application formatted file name by accessing the B-tree with reference to the operating system entry. Similarly, the multiple file name referencing system converts the application formatted file name to the operating system formatted file name by accessing the B-tree with reference to the application entry. As a result, either file name can be used to directly reference the file without requiring additional file name translation.

Brief Description of the Drawings

Figure 1 is a general block diagram of a computer system including a multiple file name referencing system in accordance with the present invention.

BNSDOCID: <EP_____0578205A2_I_>

Figure 2 is a general flow diagram of the process performed by the multiple file name referencing system of the present invention.

Figure 3 is a general flow diagram of the generating step 200 shown in Figure 2.

Figure 4 is a detailed flow diagram of the basis generation step 300 in Figure 3 as implemented to generate an application formatted file name based on a newly created operating system formatted file name.

Figure 5 is a detailed flow diagram of the conflict resolution step in Figure 3 as applied to an application formatted file name basis that has been generated from an operating system formatted file name.

Figure 6 is a detailed flow diagram of the basis generation step in Figure 3 as performed to generate an operating system formatted file name basis from an application formatted file name newly created by the user.

Figure 7 is a detailed flow diagram of the conflict resolution step 302 shown in Figure 3 as applied to an operating system formatted file name basis generated from a user created application formatted file name.

Figure 8 is a diagram of the B-tree structure in which file system entries and application entries are stored by the multiple file name referencing system of the present invention.

Figure 9 is an illustration of an application entry and an operating system entry both pointing to the same file.

Figure 10 is a flow diagram of the conversion process performed by the present invention.

Detailed Description of the Invention

Figure 1 is a general block diagram of a computer system including the multiple file name referencing system of the present invention. In Figure 1, an application 100 is provided within a computer 102 which accesses and stores various files on a peripheral device 104, such as a disk. In a preferred embodiment of the present invention, the application 100 is a DOS-based application referencing files using a "8.3" format. One of ordinary skill in the art would appreciate, however, that other types of applications using different formats could be accommodated by the generation and conversion process of the present invention. Also in the preferred embodiment, the application 100 is shown provided within the computer 102, as shown. Alternatively, the application 100 could be provided externally to the computer 102 and connected via a network.

Within the computer 102, a multiple file name referencing system 106 is provided to implement the method and system of the present invention, as will be described presently in greater detail. The multiple file name referencing system 106 is provided within a file system 108 in an operating system 110. The operating system 110 has been revised from a DOS operating system using the "8.3" format (hereinafter called "short names"), to reference operating system formatted file names of a more flexible format. This more flexible format provides for names (hereinafter called "long names") that can be up to 255 characters, and can have multiple extensions, spaces and characters considered illegal in DOS "8.3" format.

The application 100, however, was written for the previous DOS operating system which used short names and, as a result, must still refer to files using the restrictive short-name format. As a result of the revision to the operating system, the long names accessible to the operating system 110 thus differ in format from the short names which the application 100 must use.

Ordinarily under these circumstances, the application 100 would have to communicate with the operating system 110 using a short name, and could not access a more flexibly formatted file name (long name). Further, the operating system 110 could not use a long name of the more flexible format when communicating with the application 100. As will be explained, however, the present invention provides via the multiple file name referencing system 106 a method and means for referencing the same file using either the operating system formatted name (long name) or application formatted name (short name).

As will be described presently in greater detail, the multiple file name referencing system 106 serves as a generating means for generating, upon creation or renaming of a file by the application or operating system, a file name having a format different from the format of the newly provided file name. This generated file name, though having a different format, refers to the same file as the newly created file name, and preferably preserves the file extension of the newly created file name to the extent possible.

The multiple file name referencing system 106 also serves as a storing means for storing on a storage such as the disk 104, separate entries in a B-tree for file names having a different format. For example, a long name entry can be provided for a long name and a short name, both of which refer to the same file.

Additionally, the multiple file name referencing system 106 serves as a converting means for converting a file name having one format to the corresponding file name of a different format. The file name referencing system 106 also serves as a file name storing means for storing the differently formatted file

BNSDOCID: <EP_____0578205A2_I_>

names in the same file.

Figure 2 is a general flow diagram of the overall process performed by the multiple file name referencing system of the present invention. The process described in Figure 2 can be performed, for example, by the multiple file name referencing system 106. In block 200, the file name referencing system generates, upon user creation of a file name of one format, a file name of a different format. If, for example, an operating system formatted name is created, the multiple file name referencing system generates an application formatted name. Similarly, if an application formatted name is created, the multiple file name referencing system generates an operating system formatted name.

In block 202, the file name referencing system stores in a B-tree an operating system entry with the operating system formatted name and an application entry with the application formatted name. In the preferred embodiment of the invention, the operating system entry is a "long name" entry storing a long name as described above, and the application entry is a "short name" entry storing a short name of "8.3" format. In block 204, the multiple file name referencing system converts a file name having one format to the corresponding file name in the different format by accessing the B-tree. For example, a long name is converted to the corresponding short name by accessing the B-tree with reference to the long name. Alternatively, a short name can be converted to the corresponding long name by accessing the B-tree with reference to the short name.

B-tree Storage and Conversion

Figure 8 is a diagram of a B-tree structure in which the application entry (short name) and corresponding operating system entry (long name) are stored by the multiple file name referencing system in block 202 shown in Figure 2. The use of B-trees is well-known in the computer field. A detailed description of B-trees can be found in Chapter 19 of "Introduction to Algorithms," by Cormen, Leiserson and Rivest, 1990, MIT Press, pp. 381-399.

As shown in Figure 8, a B-tree is used to maintain the entries for differently formatted file names that have been generated in step 200. As is well known to one of ordinary skill in the computer field, a B-tree is a balanced search tree data structure in which data is organized in accordance with its position with reference to keys in the structure. For example, in an alphabetically arranged B-tree structure as shown in Figure 8, the letters M, D, H, etc. are keys. That is, data is divided first based on its alphabetical relationship to the letter M, then based on that to D, H, etc.

In the B-tree structure in Figure 8, file names are divided alphabetically, and thus a file name beginning with the letter "F" would be stored in the "EG" node, or in a node provided thereunder. Thus, in the present invention, if a file having the long name "filenameofgreatlength.exe" is created via the operating system 110, the multiple file name referencing system stores "filenameofgreatlength.exe" in a long name entry in the "EG" node in the B-tree.

In accordance with the present invention, when the file having the long name "filenameofgreatlength.exe" is created, a corresponding short name is automatically generated, as will be explained presently in greater detail. When the application running is a DOS application, such that the application uses file names of an "8.3" format, an "8.3" formatted short name would be generated.

For example, creation of "filenameofgreatlength.exe" would cause the multiple file name referencing system to generate a short name such as "filename.exe". If the file name "filename.exe" is already present in the B-tree, this conflict is resolved by modifying the newly generated name until it is unique, such as to "filena-1.exe". The unique, application formatted file name is then stored in an application entry in the B-tree.

As shown in Figure 9, both the operating system entry 900 and application entry 902 point to the same file 904. As a result, either short name "filena-1.exe" or long name "filenameofgreatlength.exe" can be referenced to access the same file 404. Additionally, the File Control Block in the File 904 contains both the long and short names. By storing both filenames in the File Control Block, the multiple file name referencing system allows for accessing the file through one type of formatted filename and determining the other type of formatted file name.

The multiple file name referencing system of the present invention takes advantage of the B-tree structure, such as the one described in Figure 8, to convert a file name of one format to a corresponding file name of another format, as described in block 204 in Figure 2. The multiple file name referencing system uses the known file name as a basis for searching the B-tree, comparing the known file name to the keys about which the B-tree is divided.

The B-tree search to convert the known file name to the corresponding file name (long name to short name, or vice versa) is described with reference in Figure 10. In step 1000, the file name entry for the

BNSDOCID: <EP_____0578205A2_I_>

known file name is located in the B-tree by alphabetically searching the B-tree for that name. Then, in step 1002, the leaf node containing the known file name is accessed to locate the corresponding file name. If the corresponding entry with a matching file pointer is found in step 1004, then the conversion process ends and the corresponding file name is returned.

In most cases, because of alphabetic similarity, the short name entry is located in the same leaf node as the long name entry for the same file, as shown in Figure 8. Usually, the known file name entry resides adjacent to the unknown file entry, and thus the known file name can be converted to the unknown file name by checking the adjacent entries for a file pointer that matches the file pointer of the known file name entry. When the long file name entry and short file name entry are not located adjacent to each other, then the long name entry can likely be converted to the application entry, and vice versa, by searching the entire leaf node in which the known file name entry resides.

If the entry for the corresponding name is not located in the same leaf node, then the multiple file name referencing system searches the B-tree, initially searching for a name alphabetically similar to the known file name. As a result of the overall similarity preserved between the names by the file name generation processes which produce the corresponding file names, a wildcard search allowing certain characters to vary is highly likely to produce a match. In step 1006, a wildcard is generated in which the characters in the name most likely to be changed in the generation process may vary (for example, "filena??.exe" would be generated for known file name "filenameofgreatlength.exe"). In step 1008, the wildcard is enumerated into all possible file names potentially described and entries for these file names are located. For example, "filena??.exe" would be enumerated into existing file names such as "filename.exe", "filena-1.exe", "filena-2.exe", etc. If the corresponding file name with a matching file pointer is found in step 1010, then the conversion process ends and the corresponding file name is returned. For example, filenameofgreatlength.exe is converted to filena-1.exe because its entry points to the same file and filena-1.exe is returned as the corresponding file name.

If, in step 1010, a matching file pointer is not found in any of the entries containing the wildcard enumerations, then the B-tree is searched based on an enumeration of a random wildcard "*.*" in steps 1012 and 1014, and the corresponding file name having a matching file pointer is returned.

Thus, if the short name entry and long name entry are not located in the same leaf node, one can be converted to the other by searching the entire tree with a wildcard, instead of just searching a leaf node. Such would be the case in the unlikely event that a file name is located on a different side of a key than the file name from which it is generated. For example, if filenameofgreatlength.exe and filena-2.exe pointed to the same file, and filena-1.exe were a key in the B-tree, then filenameofgreatlength.exe and filena-2.exe, as a result of being alphabetically divided about "filena-1.exe.", would be divided into separate leaf nodes. Thus, a B-tree-wide search of the wildcard "filena??.exe" would be necessary to return the entry with the address pointer to the same file as the entry for the known filename.

File Name Generation

In the preferred embodiment of the present invention, the file name generation process represented by box 200 in Figure 2 can be applied to generate an application formatted file name (short name) based on a known operation system formatted file name (long name), or vice versa. Although short names are limited to "8.3" format as explained in more detail below, long names can be any length up to 255 characters, and are not restricted by the same rules regarding illegal characters, etc.

Figure 3 is a general flow diagram of the generating block 200 by which the file name of a different format is generated. Regardless of whether a short name is generated from a long name or vice versa, the generation process can be separated into two general parts. First, in step 300, the newly created file name is used to create a basis for a file name of the different format. As will be described in more detail, this involves altering the newly created name to conform to the restrictions imposed by the different name format.

In step 302, the basis created in step 300 is modified as necessary to prevent a replication of an existing file name. When the basis coincides with an existing file name, the basis is modified until a unique file name is derived.

Figure 4 is a detailed flow diagram of the basis generation step 300 in Figure 3 as implemented to generate a short name based on a newly created long name. This generation process generates an "8.3" DOS formatted file name based on the following rules.

1. Spaces are not allowed within the file name.
2. Only one embedded period is allowed within the file name. That is, initial periods, trailing periods, and multiple periods are not allowed.

BNSDOCID: <EP_____0578205A2_1_>

3. The characters ":", " + ", "," ";", " = ", "[" and "]" are illegal and are not allowed within the file name.

4. Where a period is embedded within the file name, the "leading name" before the period is limited to eight characters and the "extension" after the period is limited to three characters.

Based on these rules, the operating system format to application format generation process performs the following modifications.

1. Spaces are removed. For example, "My File" becomes "Myfile".

2. All but the first embedded period are removed. For example, ".logon" becomes "logon", "backup.c.o" becomes "backup.co", and "main." becomes "main".

3. All illegal characters are translated to an underscore "_".

4. The leading name is truncated to eight characters and any extension is truncated to three characters.

In the flow diagram in Figure 4, in step 400, a character from the long name is retrieved, beginning with the first character. If, in step 402, the character is a space, the character does not become a character in the application formatted file name.

In step 404, if the character is a period, additional testing is performed. If, in step 406, the period is the first embedded period in the long name, it becomes a character in the short name. If the period is not the first embedded period encountered in the long name, then the period does not become a character in the short name. For example, if the period is at the very beginning or very end of the long name, or if it is the second, third, etc. period in the long name, it does not become a character in the short name. Once the first embedded period is encountered, a flag is set to indicate that following characters are in the extension portion of the file name.

In step 408, it is first determined whether the current character is part of the extension of the file name. If the extension flag has not been set (indicating that the first embedded period has not yet been encountered) then in step 410 it is determined whether the character is still within the first eight leading characters. If so, testing continues. If not, the character is not included and the next character is read in.

If the extension flag has been set (indicating encounter of the first embedded period), then in step 412 it is determined whether the current character is within the first three extension characters. If so, testing continues. If not, the character is not included and the next character is read in.

In step 414, if the character falls within the set of illegal DOS characters, an underscore character is substituted for the illegal character in step 416 and the underscore becomes a character in the short name. Otherwise, the character is maintained "as is" and becomes a character in the short name.

In step 418, characters to be included are included in the short name. To reach step 418, a character must have not violated any of the rules required of characters within DOS names, or must have been changed to conform to these rules, as explained above. In step 420, steps 400-418 are repeated as long as additional characters are present in the long name.

The long name to short name generation process illustrated in Figure 4 is more specifically defined by the following pseudocode:

BNSDOCID: <EP_____0578205A2_I_>

```
newperiod = false
leading count = 0
extensioncount = 0
extension = false
illegal = {":", "+", ",", ";", "=", "[", "]"}
while charactersinname
      if newperiod
            addcharacter
            newperiod = false
      get character
      if character <> " "
            if character <> "."
                  if extension
                        if extensioncount < = 3
                           addcharacter
                           extensioncount = extensioncount + 1

                  else if leading count < = 8
                           addcharacter
                           leadingcount = leadingcount + 1
            else if leadingcount < > 0 and not extension
                        newperiod = true
                        extension = true
```

```
addcharacter
      if character = illegal
            character = "_"
      include character
```

Figure 5 is a detailed flow diagram of the conflict resolution step 302 in Figure 3 as applied to a short name basis that has been generated from a long name. In step 500, the number of conflicts with other file names is initially set to 0. If, in step 502, no other files exist in the B-tree for which all characters are the same as the file name basis, then the process ends and the basis becomes the short name. If, however, all of the characters match an existing file name, then control proceeds to step 504, where the number of conflicts is incremented. In step 506, the last characters of the leading name are replaced with a dash "-" followed by a character or characters indicating the number of conflicts thus encountered. For example, when the first conflict has been encountered for filename "SKIPFILE.EXE", then the file name is changed to "SKIPFI-1".EXE. Control then returns to step 502, where the newly modified file name is again compared to existing file names, and steps 502-506 are repeated as long as a conflict exists. When a unique name has been derived, the process ends and the modified file name becomes the short name.

The conflict resolution process shown in Figure 5 can be performed, for example, as illustrated by the following pseudocode.

BNSDOCID: <EP_____0578205A2_I_>

```
conflicts = 0
while all characters match
      conflicts = conflicts + 1
      if conflicts < 10
            name (7) = "-"
            name (8) = char (conflicts)
      if conflicts < 100
            name (6) = "-"
            name (7,8) = char (conflicts)
      if conflicts < 1000
            .
            .
            .
```

It should be noted from the above explanation that, to the extent possible within DOS "8.3" format file name constraints, the file name extension in the operating system formatted file name is preserved in generating the application formatted file name. As a result, files of a particular file type (and thus meant to be treated a particular way) will be so treated regardless of the format with which they are referenced. For example, filenameofgreatlength.exe would be converted to filena-1.exe, such that either name denotes the file type as an executable file.

Figure 6 is a detailed flow diagram of the basis generation step 300 in Figure 3 as performed to generate a long name based on a newly created short name. In generating a long name from a short name, any sequence of characters in the short name will be legal in the long name. Thus, the short name is initially copied to become the basis for the long name. In step 600, a character, initially the first character, is obtained from the application formatted file name. In step 602, the character becomes a corresponding character in the long name basis. In step 604, step 600 and 602 are repeated as long as characters exist in the newly created short name. As a result, the newly crated short name becomes the long name basis.

Figure 7 is a detailed flow diagram of the conflict resolution step 302 shown in Figure 3 as applied to a long name basis that has been generated from a user created short name. This process is similar to the conflict resolution of a short name basis generated from a long name. Because of the more flexible format of long names, however, the conflict number can be appended to the leading name instead of replacing its final characters.

In step 700, the number of conflicts with existing file names is initially set to 0. If, in step 702, no files exist for which all characters match the file name basis, then the process ends and the file name basis becomes the long name. If, however, a conflict is determined in step 702, control proceeds to step 704, where the number of conflicts is incremented, and in step 706, the leading name is appended with a dash "-" and a character or characters indicating the number of conflicts determined thus far. For example, when a first conflict has been encountered for filename "SKIPFILE.EXE" then the file name is changed to "SKIPFILE-1.EXE". It should be noted that, unlike the short name basis in which characters must be sometimes replaced, the long name basis may simply be appended because of the greater allowed length of the name. Control then returns to step 702, and step 704 and 706 are repeated as long as file names exist which conflict with the file name basis created. When a unique file name is derived, it becomes the long name.

The conflict resolution step shown in Figure 7 can be performed, for example, as illustrated by the following pseudocode.

BNSDOCID: <EP_____0578205A2_I_>

```
conflicts = 0
while all characters match
      conflicts = conflicts + 1
      name (9) = "-"
      name (10,11,...) = char (conflicts)
```

It should be noted from the above explanation that the file name extension in the application formatted file name (short name) is entirely preserved in generating the operating system formatted file name (long name). As a result, files meant to be treated a particular way and designated as such by the extension, such as ".EXE" files, will be so treated regardless of the format with which they are referenced.

As a result of providing the multiple file name referencing system described above, either file name can be used to accurately and directly reference a file without requiring additional file name translation.

Although the present invention has been described in terms of the above-preferred embodiment, the invention itself is not limited to the embodiment described. Implementations and modifications within the spirit of the invention will be apparent to those of ordinary skill in the art. The scope of the invention itself is defined in the following claims.

Claims

1. A method performed by a computer of referencing a file, the computer having an operating system using an operating system file name format and an application using an application file name format, the method comprising the steps of:
   a) generating, upon user creation of an operating system formatted name having the operating system file name format, an application formatted name having the application file name format;
   b) storing an operating system entry for the operating system formatted name and an application entry for the application formatted name in a B-tree; and
   c) converting the operating system formatted name to the application formatted name by accessing the B-tree with reference to the operating system formatted name.

2. The method according to claim 1 wherein step a) comprises generating the application formatted name having a same file extension as the operating system formatted file name.

3. The method according to claim 2 wherein step a) comprises generating a basis derived from the operating system formatted file name and then modifying a portion of the basis other than the file extension to produce the application formatted file name.

4. The method according to claim 3 wherein step a) comprises removing all spaces from the operating system formatted file name.

5. The method according to claim 3 wherein step a) comprises removing all periods except the first period embedded within the operating system formatted file name.

6. The method according to claim 3 wherein step a) comprises replacing all predefined illegal characters with underscores.

7. The method according to claim 3 wherein step a) comprises truncating a leading name and truncating an extension within the operating system formatted file name.

8. The method according to claim 3 wherein step a) comprises modifying the leading name in the operating system formatted file name upon a conflict with an existing file name until the operating system formatted file name is unique.

9. The method according to claim 1 wherein step b) comprises storing an address of the file in both the operating system entry and the application entry in the B-tree, and wherein step c) comprises searching the B-tree, based on a location of the operating system formatted file name, for an

BNSDOCID: <EP_____0578205A2_1_>

application entry storing the address of the same file.

10. The method according to claim 9 further comprising the step of d) storing in the file both the operating system formatted file name and the application formatted file name.

11. The method according to claim 9 wherein step c) comprises searching all entries in a leaf node in which the operating system entry is stored.

12. The method according to claim 9 wherein step c) comprises searching all entries in the B-tree having common characters with the operating system formatted name.

13. The method according to claim 9 wherein step c) comprises random searching all entries in the B-tree until the address of the file as stored in the operating system entry is located.

14. A method performed by a computer of referencing a file, the computer having an operating system using an operating system file name format and an application using an application file name format, the method comprising the steps of:
a) generating, upon user creation of an application formatted file name having the application file name format, an operating system formatted file name having the operating system file name format;
b) storing an operating system entry for the operating system formatted name and an application entry for the application formatted name in a B-tree; and
c) converting the application formatted name to the operating system formatted name by accessing the B-tree with reference to the application formatted file name.

15. The method according to claim 14 wherein step a) comprises generating the operating system formatted file name having a same file extension as the application formatted file name.

16. The method according to claim 15 wherein step a) comprises generating a basis derived from the application formatted file name and then modifying a portion of the basis other than the file extension to produce the operating system formatted file name.

17. The method according to claim 16 wherein step a) comprises setting the operating system file name basis equal to the application formatted file name.

18. The method according to claim 17 wherein step a) comprises appending a leading name in the application formatted file name upon a conflict with an existing file name until the application-formatted file name is unique.

19. The method according to claim 14 wherein step b) comprises storing an address of the file in both the operating system entry and the application entry in the B-tree, and wherein step c) comprises searching the B-tree, based on a location of the application formatted file name, for an operating system entry storing the address of the same file name.

20. The method according to claim 19 further comprising the step of d) storing in the file both the operating system formatted file name and the application formatted file name.

21. The method according to claim 19 wherein step c) comprises searching all entries in a leaf node in which the application entry is stored.

22. The method according to claim 19 wherein step c) comprises searching all entries in the B-tree having common characters with the application formatted named in accordance with the generation performed in step (a).

23. The method according to claim 19 wherein step c) comprises random searching all entries in the B-tree until the address of the file as stored in the application entry is located.

24. A multiple file name referencing system for referencing a file stored by a computer, the computer having an operating system using an operating system file name format and an application using an

BNSDOCID: <EP_____0578205A2_I_>

application file name format, the multiple file name referencing system comprising:

generating means for generating, upon user creation of an operating system formatted file name having the operating system file name format, an application formatted file name having the application file name format;

storing means for storing an operating system entry for the operating system formatted file name and an application entry for the application formatted file name in a B-tree; and

converting means for converting the operating system formatted file name to the application formatted file name by accessing the B-tree with reference to the operating system formatted file name.

25. The multiple file name referencing system according to claim 24 wherein the generating means generates the application formatted file name having a same file extension as the operating system formatted file name.

26. The multiple file name referencing system according to claim 25 wherein the generating means generates a basis derived from the operating system formatted file name and then modifies a portion of the basis other than the file extension to produce the application formatted file name.

27. The multiple file name referencing system according to claim 26 wherein the generating means removes all spaces from the operating system formatted file name, removes all periods except the first period embedded within the operating system formatted file name, replaces all predefined illegal characters with underscores and truncates a leading name and an extension within the operating system formatted file name.

28. The multiple file name referencing system according to claim 27 wherein the generating means modifies the leading name in the operating system formatted file name upon a conflict with an existing file name until the operating system formatted file name is unique.

29. The multiple file name referencing system according to claim 24 wherein the storing means stores an address of the file in both the operating system entry and the application entry in the B-tree, and wherein the converting means searches the B-tree based on a location of the operating system formatted file name for an application entry storing the address of the same file.

30. The multiple file name referencing system according to claim 29 further comprising file name storing means for storing in the file both the operating system formatted file name and the application formatted file name.

31. The multiple file name referencing system according to claim 29 wherein the converting means searches all entries in a leaf node in which the operating system entry is stored.

32. The multiple file name referencing system according to claim 29 wherein the converting means searches all entries in the B-tree having common characters with the operating system formatted file name.

33. The multiple file name referencing system according to claim 29 wherein the converting means random searches all entries in the B-tree until the address of the file as stored in the operating system entry is located.

34. A multiple file name referencing system for referencing a file stored by a computer, the computer having an operating system using an operating system file name format and an application using an application file name format, the multiple file name referencing system comprising:

generating means for generating, upon creation of an application formatted file name having the application file name format, an operating system formatted file name having the operating system file name format;

storing means for storing an operating system entry for the operating system formatted file name and an application entry for the application formatted file name in a B-tree; and

converting means for converting the application formatted name to the operating system formatted name by accessing the B-tree with reference to the application formatted name.

BNSDOCID: <EP_____0578205A2_1_>

35. The multiple file name referencing system according to claim 34 wherein the generating means generates the operating system formatted name having a same file extension as the application formatted file name.

36. The multiple file name referencing system according to claim 35 wherein the generating means generates a basis derived from the application formatted name and then modifies a portion of the basis other than the file extension to produce the operating system formatted file name.

37. The multiple file name referencing system according to claim 35 wherein the generating means sets the basis equal to the application formatted name and then appends a leading name in the application formatted file name to produce the operating system file name.

38. The multiple file name referencing system according to claim 34 wherein the storing means stores an address of the file in both the operating system entry and the application entry in the B-tree, and wherein the converting means searches the B-tree, based on a location of the application formatted file name, for an operating system entry storing the address of the same file.

39. The multiple file name referencing system according to claim 38 further comprising file name storing means for storing in the file both the operating system formatted file name and the application formatted file name.

40. The multiple file name referencing system according to claim 38 wherein the converting means searches all entries in a leaf node in which the application entry is stored.

41. The multiple file name referencing system according to claim 38 wherein the converting means searches all entries in the B-tree having common characters with the application formatted file name.

42. The multiple file name referencing system according to claim 38 wherein the converting means random searches all entries in the B-tree until the address of the file as stored in the application entry is located.

BNSDOCID: <EP_____0578205A2_I_>

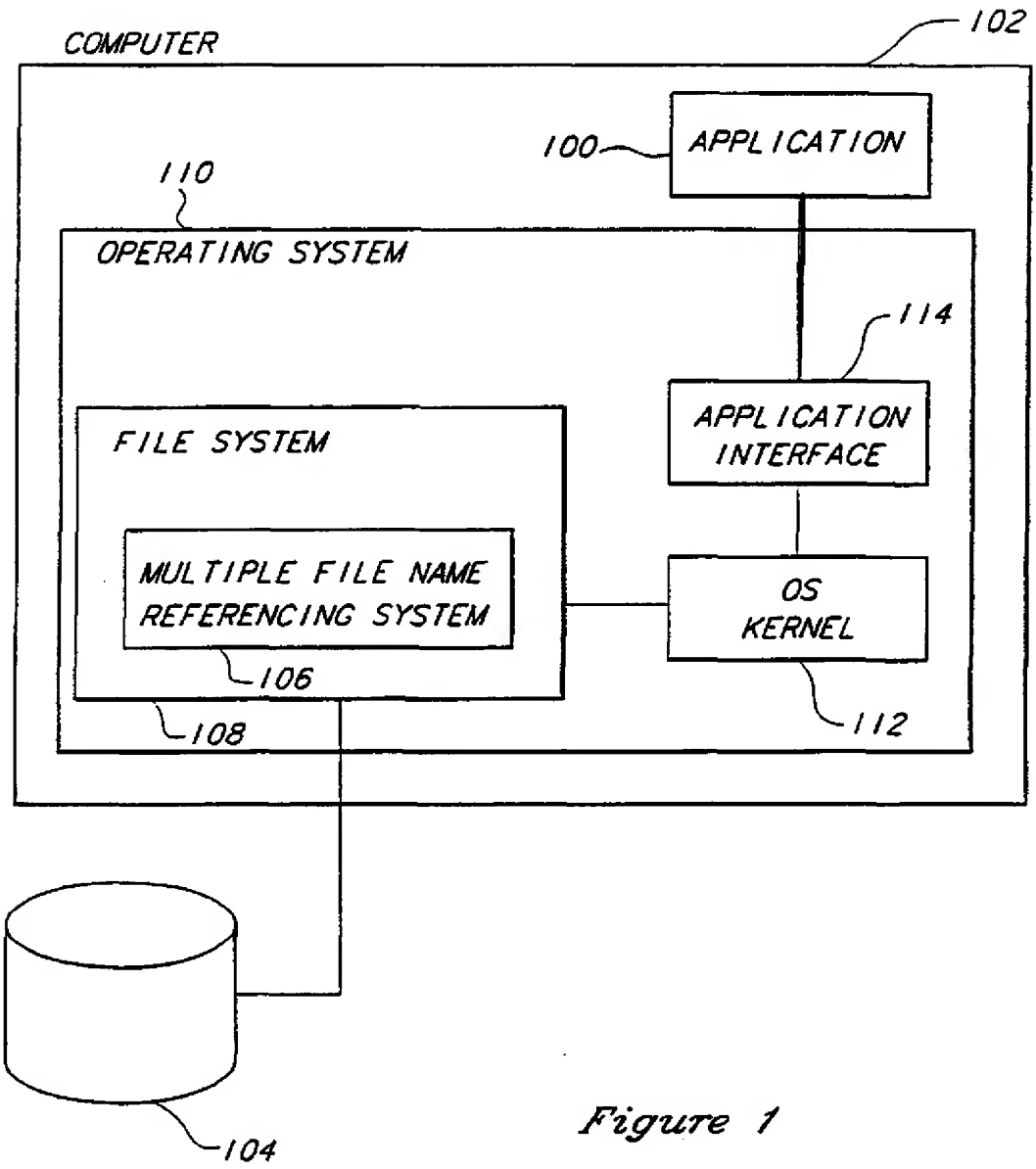


Figure 1

BNSDOCID: <EP_____0578205A2_I_>

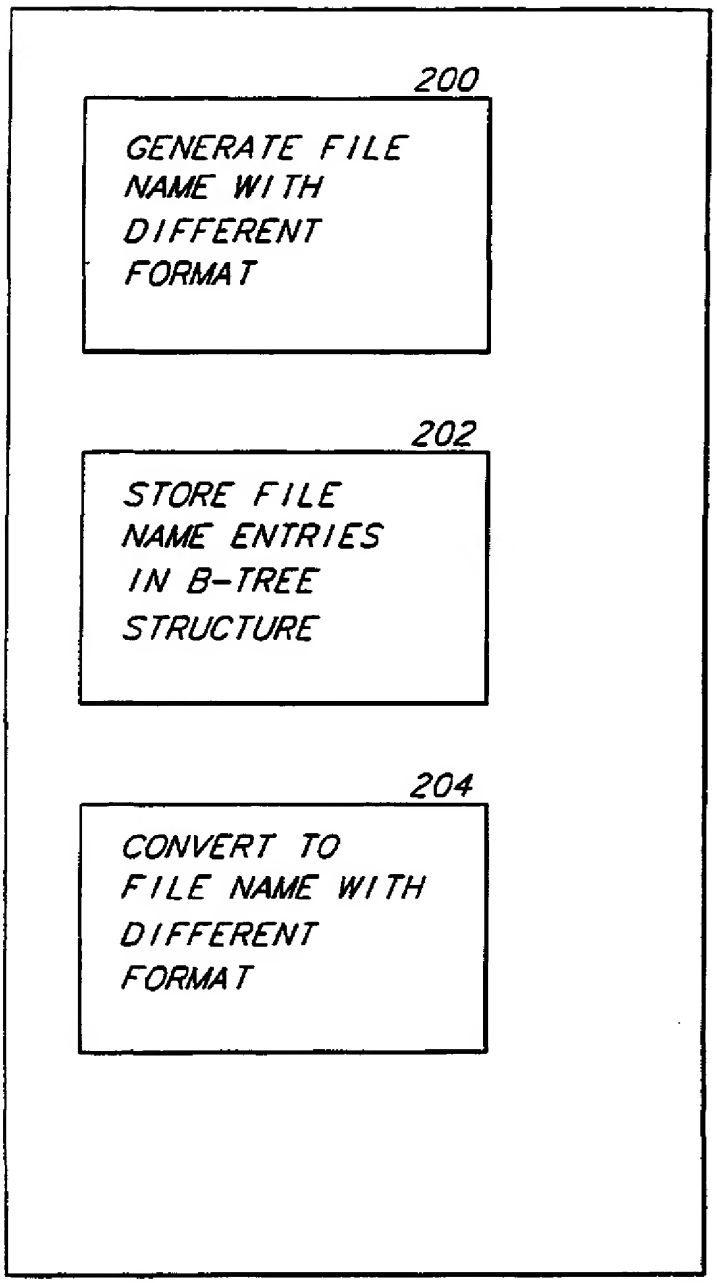


Figure 2

BNSDOCID: <EP_____0578205A2_I_>

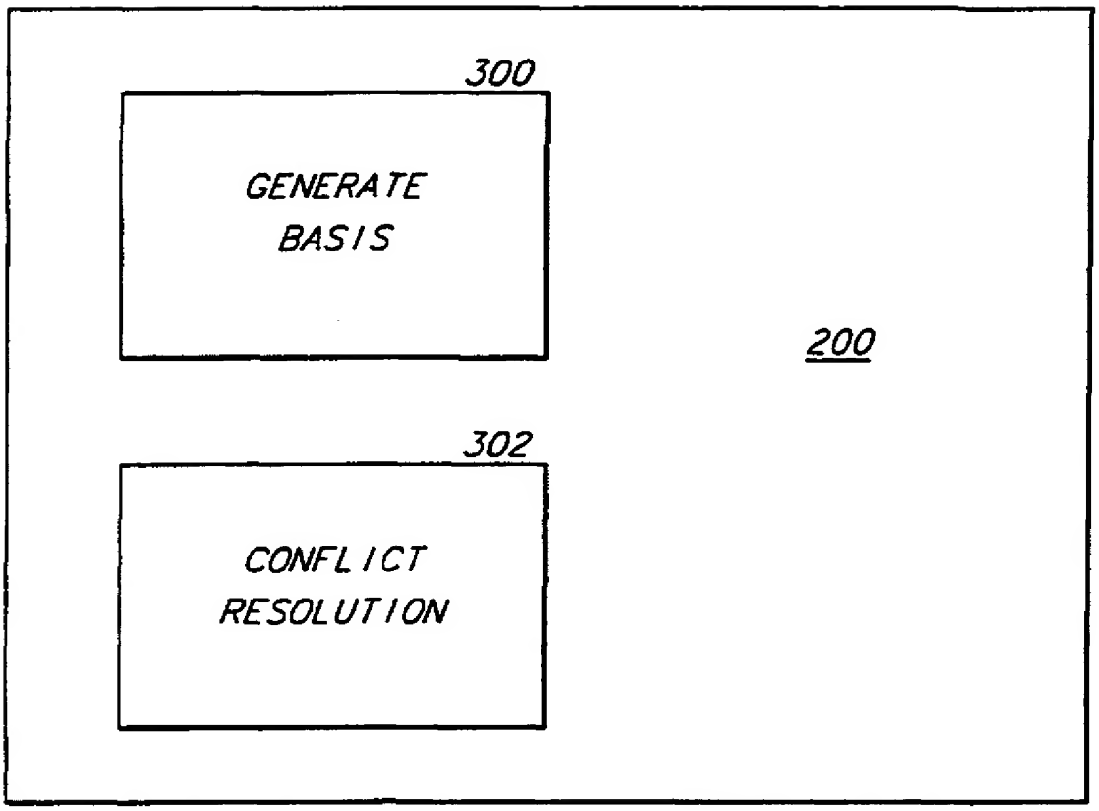


Figure 3

BNSDOCID: <EP_____0578205A2_I_>

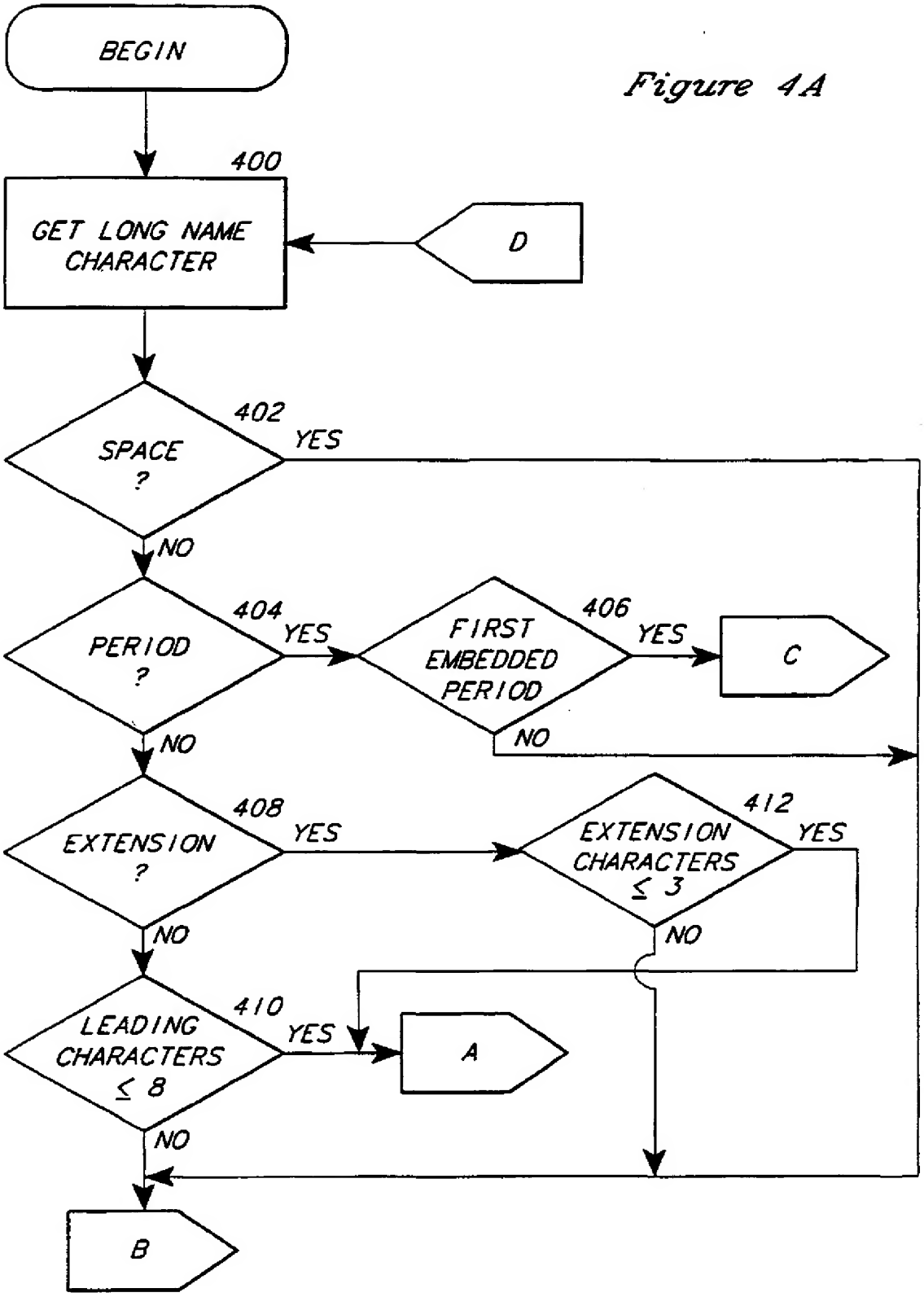
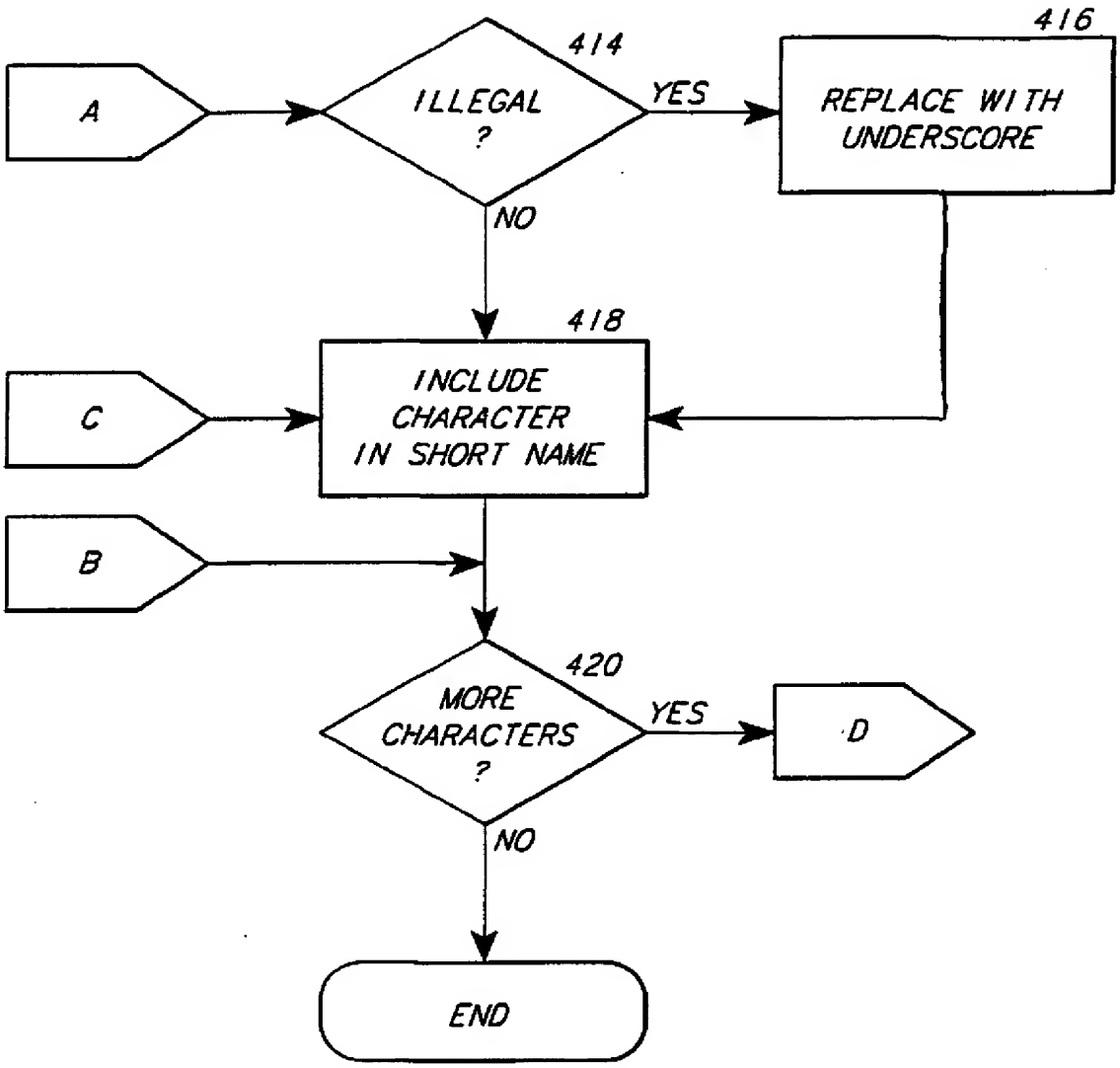


Figure 4A

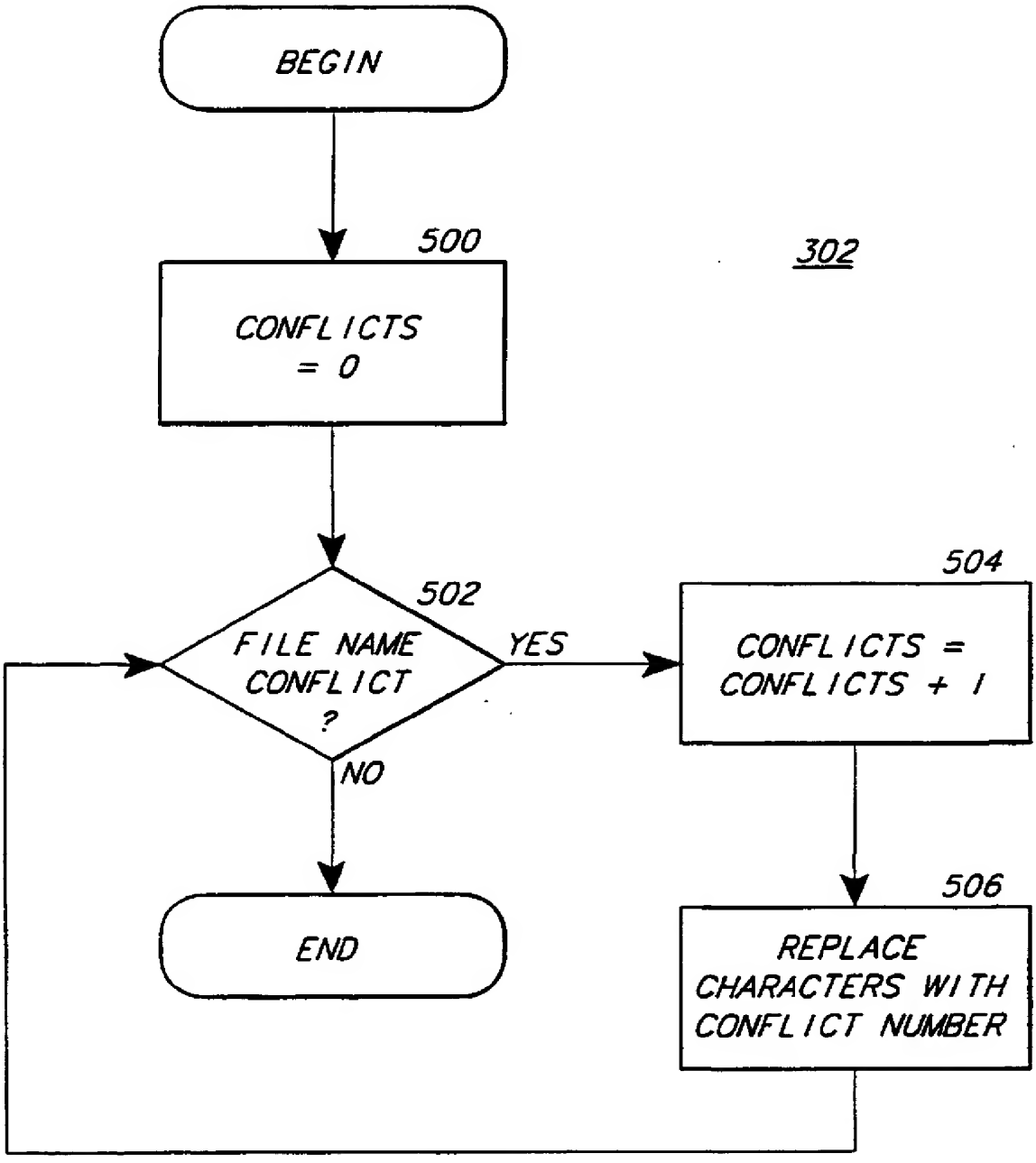
BNSDOCID: <EP_____0578205A2_I_>

Figure 4B



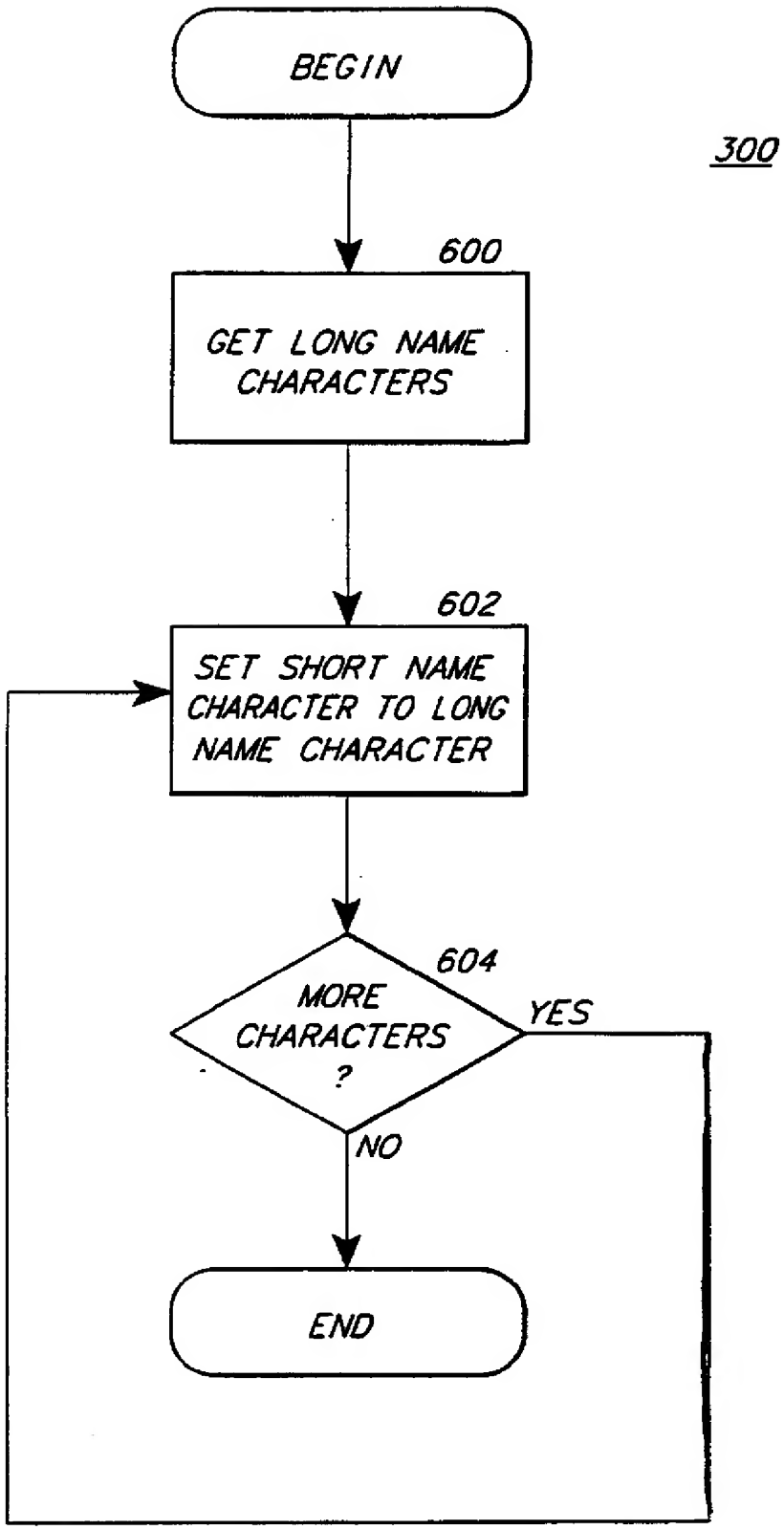
BNSDOCID: <EP_____0578205A2_I_>

Figure 5



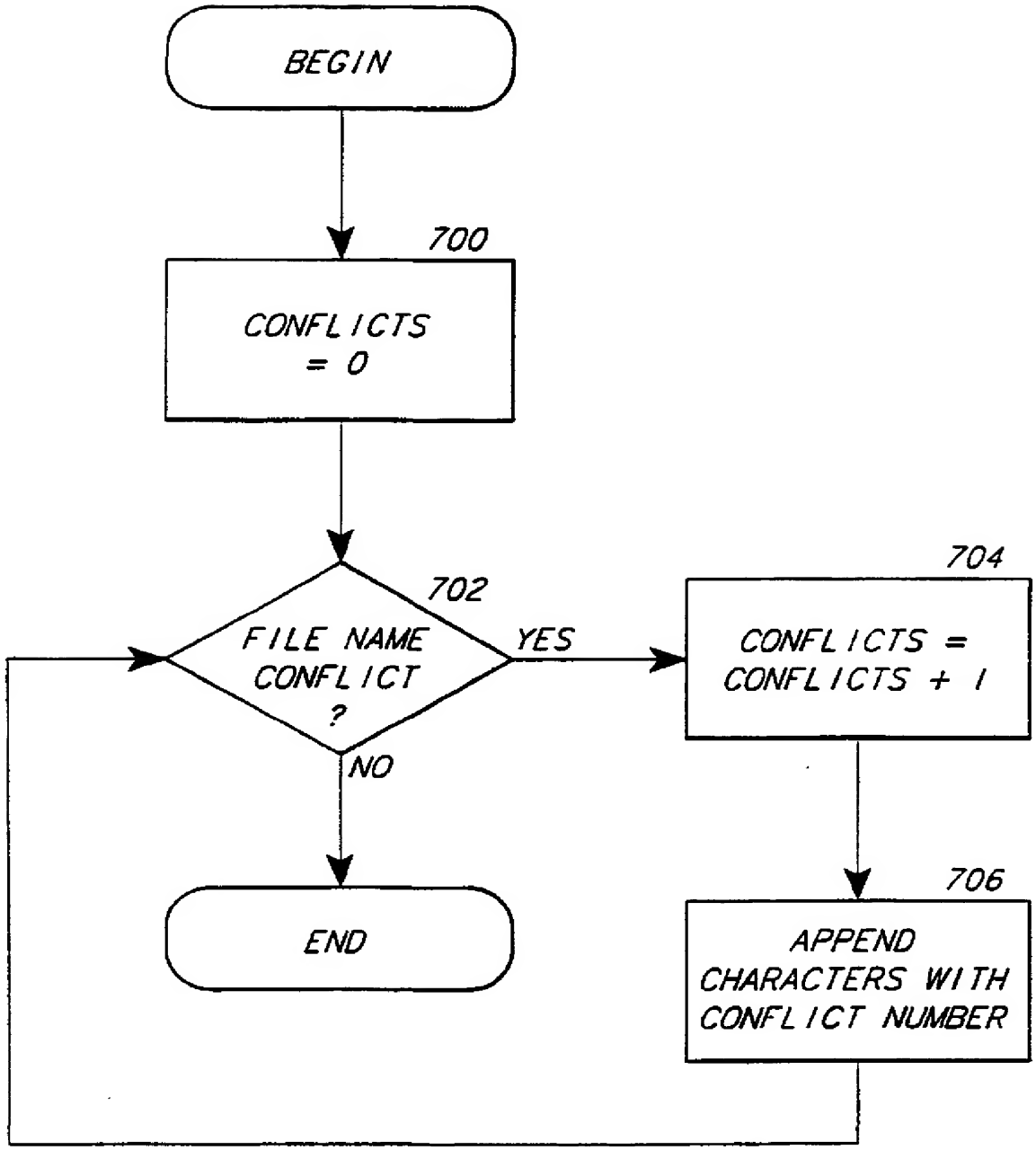
BNSDOCID: <EP_____0578205A2_1_>

Figure 6



BNSDOCID: <EP_____0578205A2_I_>

Figure 7



BNSDOCID: <EP_____0578205A2_I_>

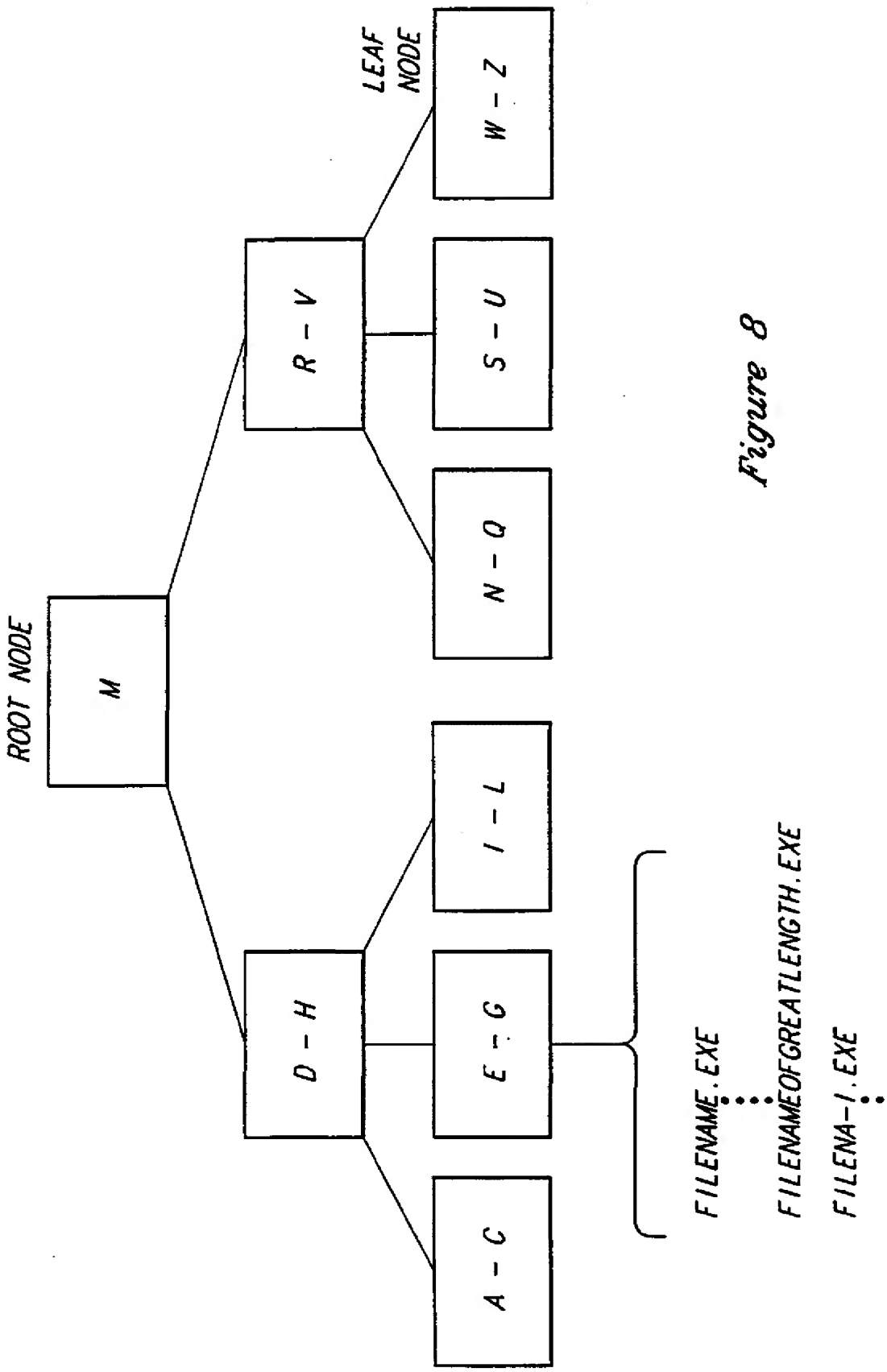


Figure 8

BNSDOCID: <EP_____0578205A2_I_>

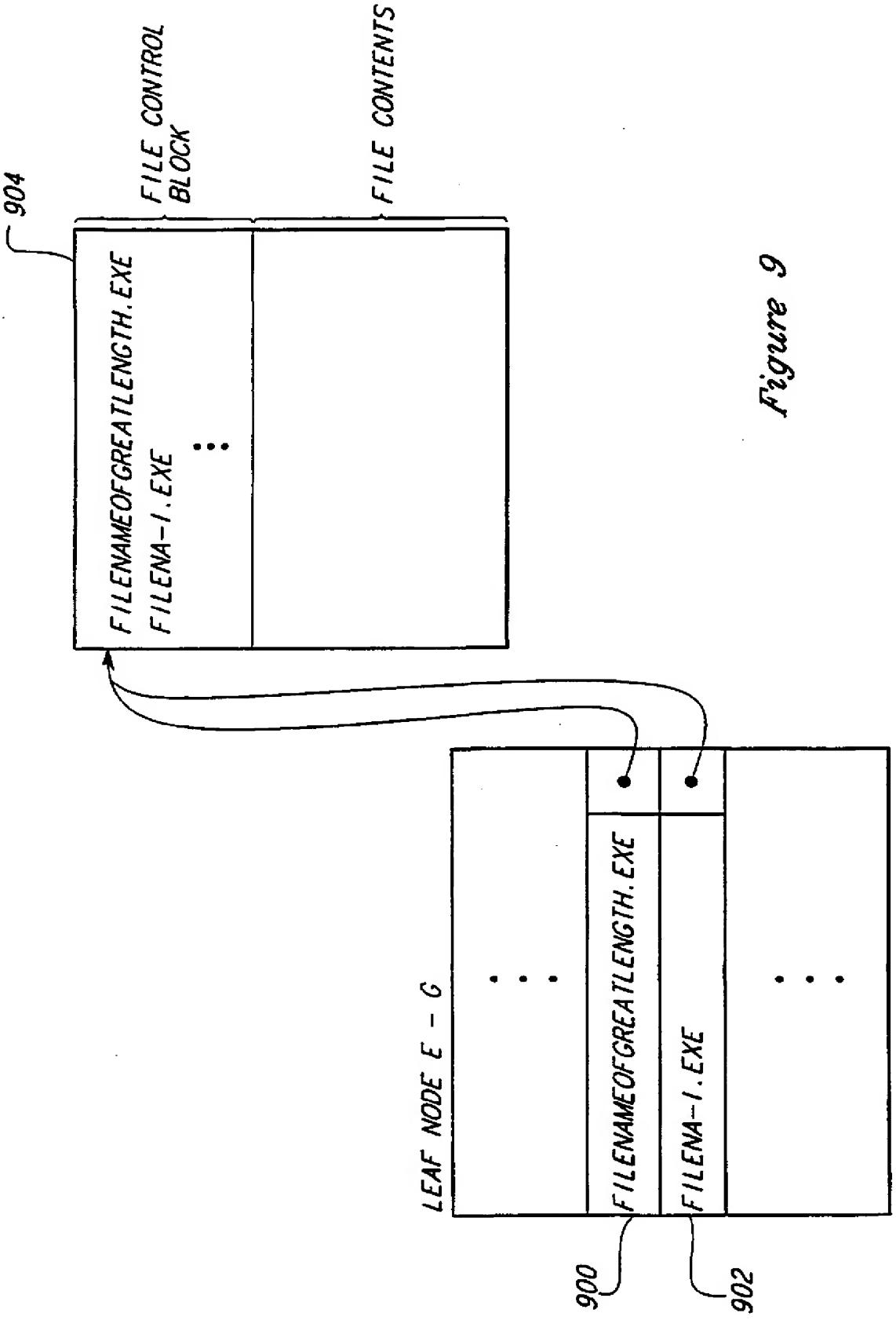


Figure 9

BNSDOCID: <EP_____0578205A2_I_>

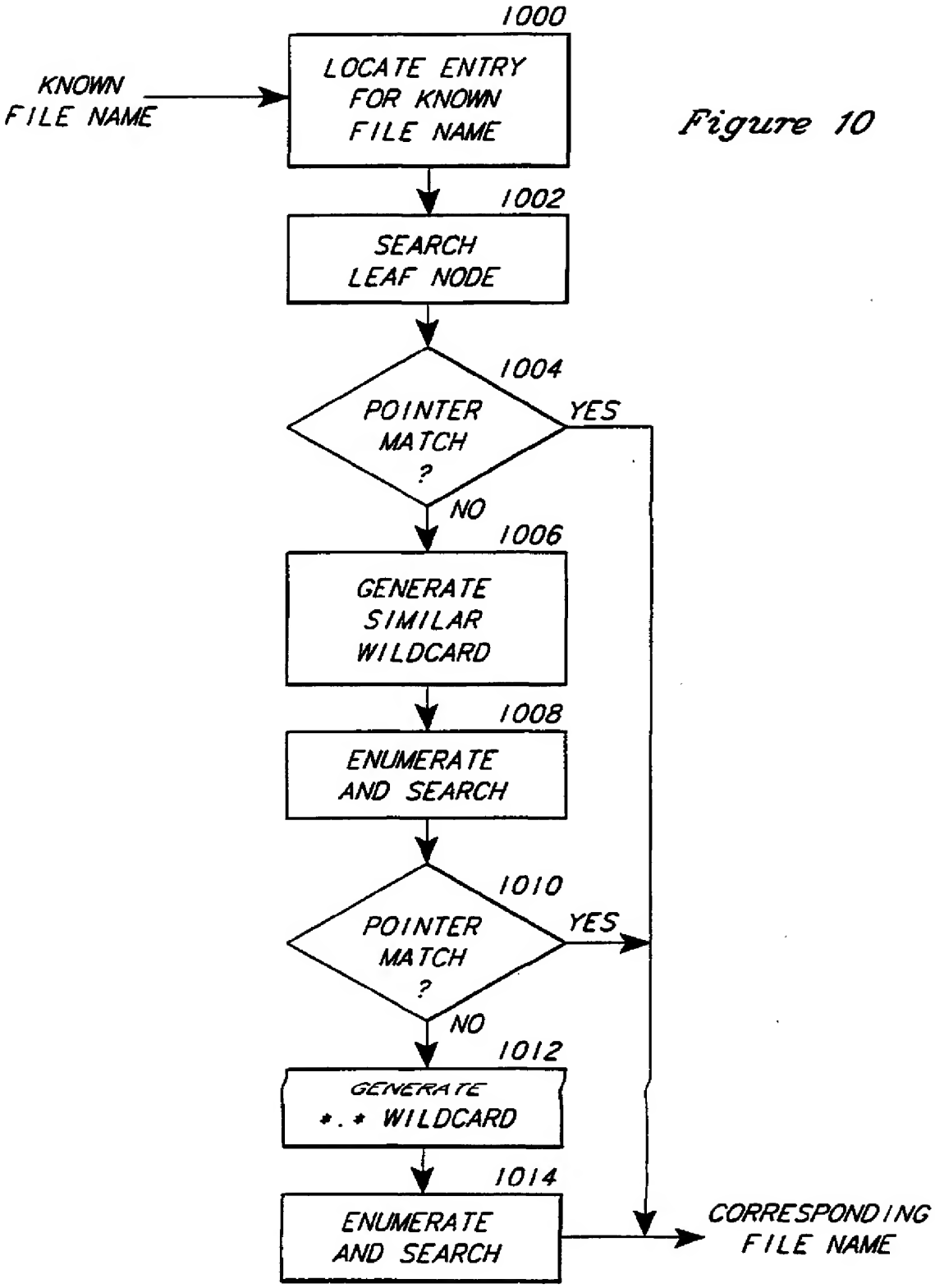


Figure 10

BNSDOCID: <EP_____0578205A2_I_>

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 578 205 A3**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **93110797.3**

(51) Int. Cl.[5]: **G06F 15/403**

(22) Date of filing: **06.07.93**

(30) Priority: **06.07.92 US 909984**

(43) Date of publication of application:
12.01.94 Bulletin 94/02

(84) Designated Contracting States:
AT BE CH DE DK ES FR GB GR IE IT LI LU MC NL PT SE

(88) Date of deferred publication of the search report:
18.05.94 Bulletin 94/20

(71) Applicant: **MICROSOFT CORPORATION**
One Microsoft Way
Redmond, Washington 98052-6399(US)

(72) Inventor: **Miller, Thomas J.**
3135 Fairweather Place
Bellevue, Washington 98004(US)
Inventor: **Kimura, Gary D.**
6807 126th Avenue N.E.
Kirkland, Washington 98033(US)

(74) Representative: **Patentanwälte Grünecker, Kinkeldey, Stockmair & Partner**
Maximilianstrasse 58
D-80538 München (DE)

(54) Multiple file name referencing system.

(57) A multiple file name referencing system stores multiple file names in a file. These multiple file names include an operating system formatted file name and an application formatted file name. When an operating system formatted file name is created or renamed, the multiple file name referencing system automatically generates an application formatted file name having a potentially different format from, but preserving the extension of, the operating system formatted name. The multiple file name referencing system similarly generates an operating system formatted name upon creation or renaming of an application formatted name. A B-tree is provided which contains an operating system entry for the operating system formatted name and an application entry for the application formatted name, each entry containing the address of the same file to which both names refer. The multiple file name referencing system converts the operating system formatted file name to the application formatted file name by accessing the B-tree with reference to the operating system entry, and vice versa. As a result, either file name can be used to directly reference the file without requiring additional file name translation.

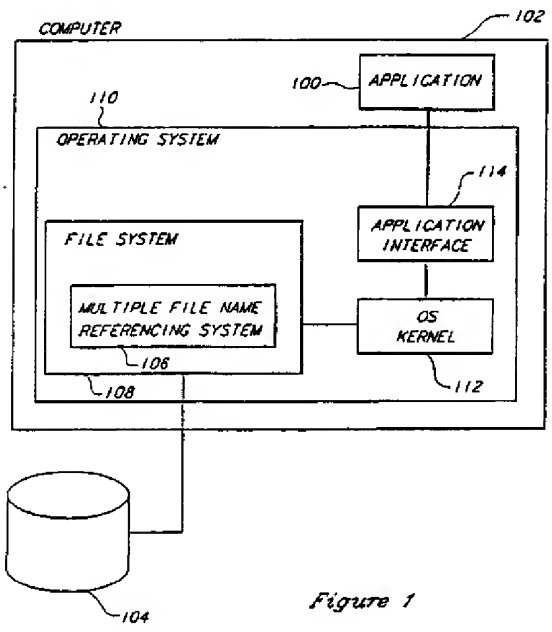


Figure 1

EP 0 578 205 A3

BNSDOCID: <EP_____0578205A3_I_>

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 93 11 0797

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.5)
X	IEEE 6TH INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS 23 May 1986 , CAMBRIDGE, MASSACHUSETTS, US pages 509 - 514 D. COMER ET AL. 'The Tilde File Naming Scheme' * the whole document *	1,14,24, 34	G06F15/403
			TECHNICAL FIELDS SEARCHED (Int.Cl.5) G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	25 January 1994	Katerbau, R

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

BNSDOCID: <EP_____0578205A3_I_>